Programming FPGA EEPROM Configuration Memory Over VME

:

Using the FPGA to Configure Its Own EEPROM Configuration Memory

Craig Drennan
December 16, 2004

I. Introduction

The modules being developed for the Beam Loss Monitor Upgrade are 6U VME module utilizing large FPGA's to implement tasks such as data transfer, abort decision logic and signal processing. The desire is to be able to download the FPGA configuration information via the VME crate processor and VME bus to the serial EEPROM devices that program the FPGA at power up. The approach retains the ability to use an EEPROM programming cable connected directly to the PCB programming header as is done in development.

This note will describe how the circuits are wired and list the documentation that the programmer will need to understand how to format and load the configuration files. Since this approach is not specifically prescribed in any of the FPGA vendor or EEPROM vendor literature, we will also list and explain the function and operation of all of the configuration control lines involved in programming the EEPROM and configuring the FPGA.

This document is not at all final at this point. It was written to help work out the details of programming the FPGA's with the board designers on the project.

II. The Schematic Wiring

The method's described in this note assume the use of Altera Cyclone FPGA's and Atmel AT17LV002A serial configuration memories on the BLM Upgrade VME modules..

Figure II.1 is the schematic of the connections between the FPGA and the configuration memory. The FPGA's are programmed in what Altera calls the Passive Serial (PS) mode. In this mode the FPGA never drives the DCLK or DATA0 lines. The AT17LV configurator drives these lines during FPGA configuration. During configurator programming the SER_EN / line is pulled low. When the AT17LV is in this mode it will not drive the DCLK line, but will pull the DATA line low periodically to signal the ACK acknowledge during programming.

There are two methods for programming the configurator. The first is by connecting the Atmel ADTH2225 programming cable to the ISP Header. Configuration files are downloaded to the AT17LV from a PC running Atmel's Configurator Programming System (CPS) software. The ISP Header could be mounted to the faceplate of the VME module so the configurator can be programmed without removing it from the crate. The programming cable may be connected, new configurator data downloaded and the cable disconnected while the module is running without interfering with the operation of the FPGA. In order to have the new configuration loaded into the FPGA the power to the module would need to be cycled or the nCONFIG pin on the FPGA pulled low momentarily. Note that reconfiguration will not proceed if the programming cable is connected, since it pulls the SER_EN / signal low disabling the AT17LV's DCLK output. The CPS software can be downloaded from www.atmel.com.

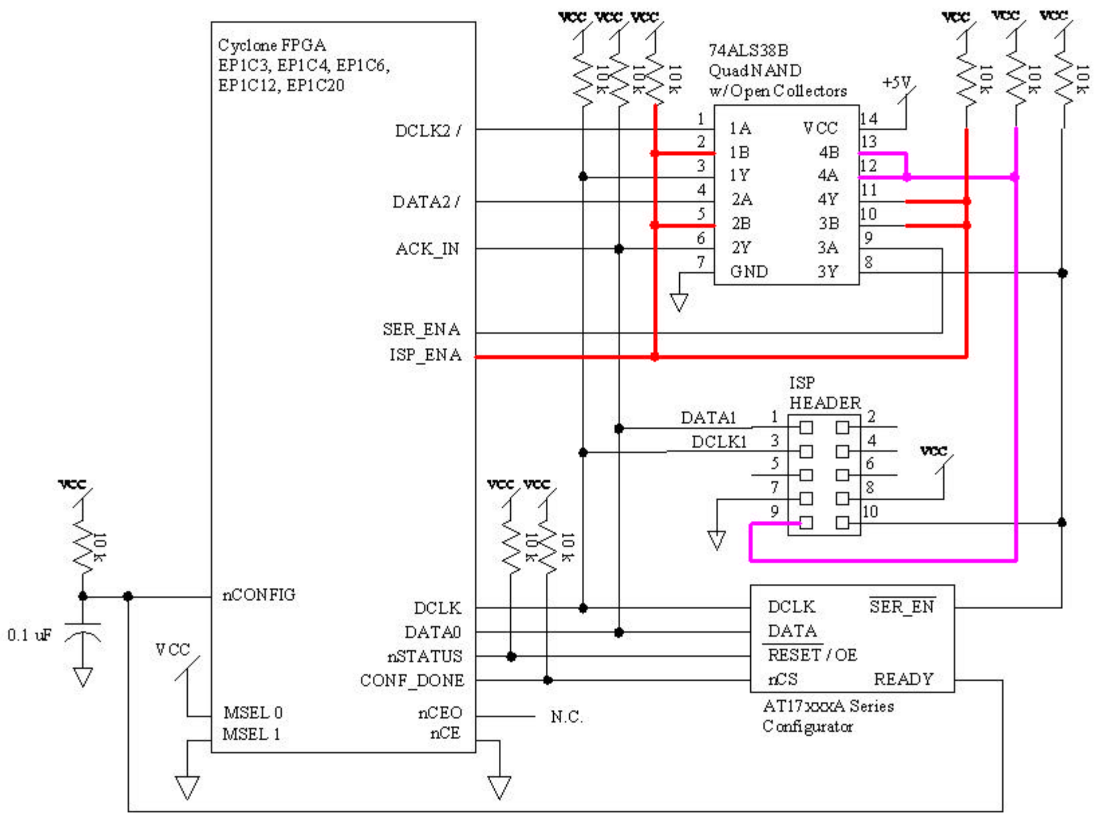


Figure II.1 The proposed hook up of the FPGA and configurator for in system programming.

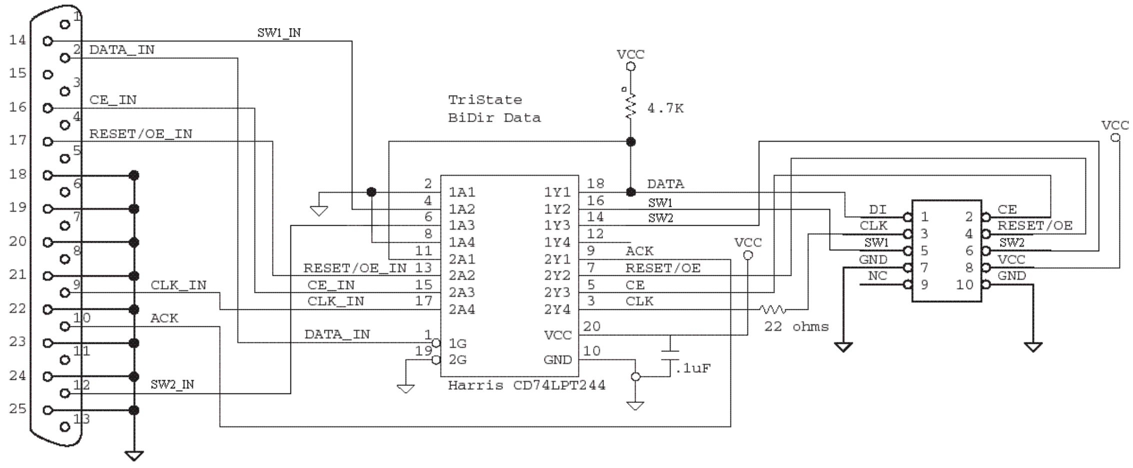


Figure II.2 The schematic of the ADTH2225 programming cable.

The second method for programming the configurator is over the VME bus, through control logic that will be programmed into the FPGA. The FPGA will drive the DATA and DCLK lines and pull down the SER_EN / pin of the AT17LV. There are a couple of concerns that are addressed in the circuit shown in Figure II.1. First, we want to avoid any situation where the FPGA and the ADTH2225 programming cable are driving the same signals at the same time. Second we must guarantee that if the control lines from the FPGA are ever mis-configured it will not interfere with our ability to program the configurator using the programming cable.

The 74ALS38B Quad 2 Input NAND Gates with open collectors is used to isolate the FPGA from the control lines. Open drain or tri-state outputs could have been implemented in the FPGA, but if these pins were accidentally reconfigured it could cause a problem. In order to enable the FPGA control signals Pin 9 needs to be jumpered to Pin 7 (GND) of the ISP Header. If a jumper shunt or another plug inserted into the ISP Header is used for this purpose it would also be a physical obstruction to connecting the programming cable. When this jumper is removed and the programming cable connected, Pin 9 will not be pulled down and the signal line on Pin 9 is set high. Figure II.3 shows how Pin 9 is used to enable or hold off the FPGA control signals using the NAND gates.

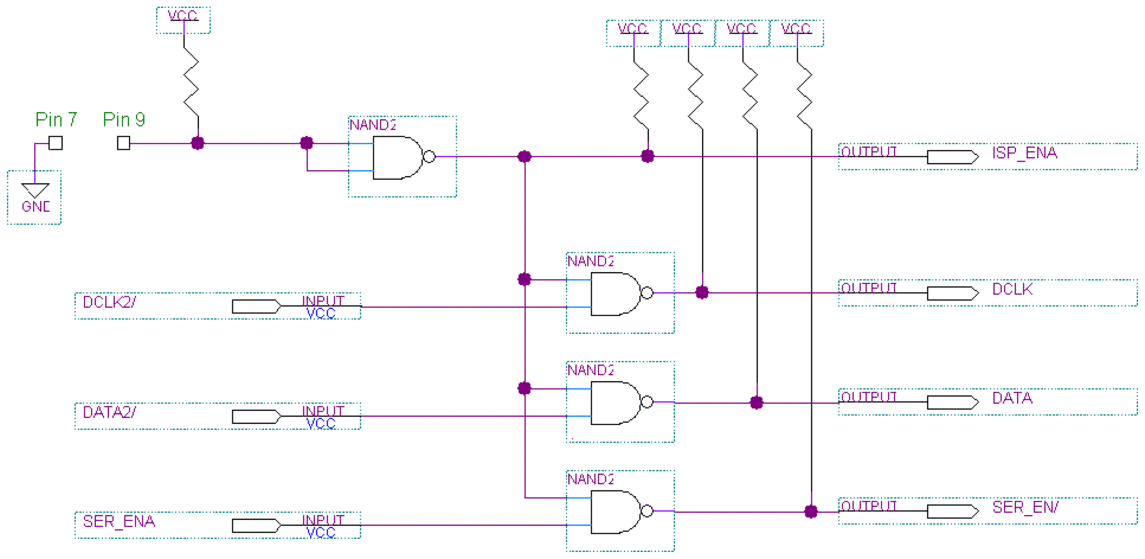


Figure II.3 Logic for isolating the FPGA configurator control signals.

An alternate way of isolating the two methods of programming was suggested. This was to connect the FPGA control signals to a second programming header which could be jumpered to the main ISP Header using a small ribbon cable assembly. This would work fine except it would preclude mounting of the ISP Header on the VME front panel since there is likely not enough room on the front panel for two headers.

III. Programming Specifications

The essential document that describes how to program the Atmel configuration memory is

"Programming Specification for AT17LV(A) Series FPGA Configuration Memories", AT17LV Series FPGA Configuration Memory Application Note, Revision 0437K-CNFG-05/03.

This document describes programming the device, verifying the device, and setting the reset polarity. It also contains the DC and AC Timing specification for the device. The other documents that should be reviewed to have a full understanding on using the AT17LV(A) configurator are

"In-System Programming Cable, ATDH2225", Revision 2288A-05/01

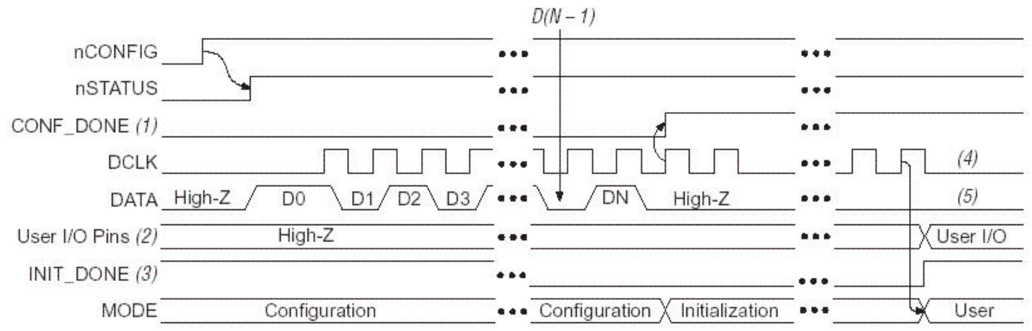
"Introducing Atmel AT17LV Series FPGA Configuration Memories" 2295B-CNFG–09/02

These documents are available on the Atmel www site.
http://www.atmel.com/products/Config/

Excerpt From Altera Documentation
"Chapter 13 Configuring Cyclone FPGAs", pages 13-15 to 13-17
http://www.altera.com/literature/hb/cyc/cyc_c51013.pdf

Passive Serial Configuration
Cyclone FPGAs also feature the PS configuration scheme supported by all Altera FPGAs. In the PS scheme, an external host (configuration device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Cyclone FPGAs via the DATA0 pin at each rising edge of DCLK. The configuration waveforms for this scheme are shown in Figure 13–8.

Figure 13–8. PS Configuration Cycle Waveform



Notes to Figure 13–8:

(1) During initial power up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high to indicate successful configuration. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.

(2) User I/O pins are tri-stated during configuration. Cyclone FPGAs also have a weak pull-up resistor on I/O pins during configuration. After initialization, the user I/O pins perform the function assigned in the user's design.

(3) When used, the optional INIT_DONE signal is high when nCONFIG is low before configuration and during the first 136 clock cycles of configuration.

(4) In user mode, DCLK should be driven high or low when using the PS configuration scheme. When using the AS configuration scheme, DCLK is a Cyclone output pin and should not be driven externally.

(5) In user mode, DATA0 should be driven high or low.

PS Configuration using Configuration Device

In the PS configuration device scheme, nCONFIG is usually tied to VCC (when using EPC16, EPC8, EPC4, or EPC2 devices, you can connect nCONFIG to nINIT_CONF). Upon device power-up, the target Cyclone FPGA senses the low-to-high transition on nCONFIG and initiates configuration. The target device then drives the open-drain CONF_DONE pin low, which in-turn drives the configuration device's nCS pin low. When exiting POR, both the target and configuration device release the open-drain nSTATUS pin (typically Cyclone POR lasts 100 ms).

Before configuration begins, the configuration device goes through a POR delay of up to 100 ms (maximum) to allow the power supply to stabilize. You must power the Cyclone FPGA before or during the POR time of the enhanced configuration device. During POR, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When the target and configuration devices complete POR, they both release the nSTATUS to OE line, which is then pulled high by a pull-up resistor.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. When all devices are ready, the configuration device clocks out DATA and DCLK to the target devices using an internal oscillator.

After successful configuration, the Cyclone FPGA starts initialization using the 10-MHz internal oscillator as the reference clock. The

CONF_DONE pin is released by the target device and then pulled high by a pull-up resistor. When initialization is complete, the target Cyclone FPGA enters user mode.

If an error occurs during configuration, the target device drives its nSTATUS pin low, resetting itself internally and resetting the configuration device. If you turn on the **Auto-Restart Configuration on Frame Error** option, the device reconfigures automatically if an error occurs. To set this option, select **Compiler Settings** (Processing menu), and click on the **Chips & Devices** tab. Select **Device & Pin Options**, and click on the **Configuration** tab.
If the **Auto-Restart Configuration on Frame Error** option is turned off, the external system (configuration device or microprocessor) must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to VCC. When configuration is complete, the target device releases CONF_DONE, which disables the configuration device by driving nCS high. The configuration device drives DCLK low before and after configuration.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the target device has not configured successfully. (For CONF_DONE to reach a high state, enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit. EPC2 devices wait for 16 DCLK cycles.) In this case, the configuration device pulses its OE pin low for a few microseconds, driving the target device's nSTATUS pin low. If the **Auto-Restart Configuration on Frame Error** option is set in the Quartus II software, the target device resets and then releases its nSTATUS pin after a reset timeout period. When nSTATUS returns high, the configuration device reconfigures the target device.

You should not pull CONF_DONE low to delay initialization. Instead, use the Quartus II software's **User-Supplied Start-Up Clock** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together since their CONF_DONE pins are tied together. For more information on this option, see "Device Options" on page 13–45.
CONF_DONE goes high during the first few clock cycles of initialization. Hence when using the CLKUSR feature you would not see the CONF_DONE signal high until you start clocking CLKUSR. However, the device does retain configuration data and waits for these initialization clocks to release CONF_DONE and go into user mode.

NOTE: When using internal pull-up resistors on configuration devices, power the supply voltage on the Cyclone FPGA I/O pins

(VCCIO) to 3.3-V. EPC2, EPC4, EPC8, and EPC16 devices support
3.3-V operation but not 2.5-V operation. Therefore, you must set
the VCCIO voltages for the banks where programming pins reside
to 3.3 V.